

The ORA-01555 Error

By Boris Milrud

The "ORA-01555 snapshot too old" error, which causes user transactions to fail, is a nightmare for Oracle DBAs and developers. It usually occurs after queries or batch processes have been running for a long time, which means you can lose many hours of processing when the error crops up.

It is hard to recreate this error and it occurs inconsistently: you may never encounter this error during the next run. The [Oracle Server Error Messages manual](#) gives the cause for this error as "Rollback segment too small." However, in most cases you'll find that your rollback segments are just fine; you have plenty of space in the rollback tablespace as well in the particular rollback segment that accommodates failed transactions.

So what is the problem and how do you solve it?

There are three situations that can cause the ORA-01555 error:

- An active database with an insufficient number of small-sized rollback segments
- A rollback segment corruption that prevents a consistent read requested by the query
- A fetch across commits while your cursor is open

The cause of the error involves a combination of the following factors: the setup of your rollback segments and the number and nature of the database concurrent transactions.

The Problem

How do you avoid the ORA-01555 error, which is hard to recreate and crops up inconsistently?

The Solution

From the DBA's perspective, you can avoid this error by allocating rollback segments and configuring them with the proper parameter values, as described.

Understanding Rollback Segments

Before delving into this problem, let me first describe rollback segments in general. Rollback segments are involved in every transaction that occurs within a database. Since they control the database's ability to handle transactions, they play a key role in the database's successful operation.

Rollback segments capture the data image as it existed prior to the start of the transaction. Other users' queries against the data that is being changed will return the data as it existed before the change began. This feature allows the database to maintain read consistency between multiple transactions. The number and size of rollback segments available are specified by the DBA during the database's creation. Information about rollback segments status is accessible via the `DBA_ROLLBACK_SEGS` view.

A rollback segment consists of contiguous multi-block pieces called *extents*. In an ideal database, each transaction fits within a single extent. However, this is rarely the case. When a transaction can no longer acquire space within an extent, the rollback segment looks for another extent to which it can continue writing the rollback segment entry. The segment uses these extents in an ordered, circular fashion, moving from one to the next after the current extent is full. A transaction writes a record to the current location in the rollback segment and advances the current pointer by the size of the record.

The ORA-01555 Error

By Boris Milrud

The current writing location for records is the *head* of the rollback segment. The term *tail* is used to refer to the location on the rollback segment that is the beginning of the oldest active transaction record.

It's also important to make sure that individual rollback segments are large enough to handle their transaction load. Oracle allocates rollback segments in a round-robin fashion among all online rollback segments (with the exception of SYSTEM) to try to spread transactions evenly.

Here are some simple rollback segment rules, per Oracle's documentation:

- A transaction can only use one rollback segment to store all of its undo records.
- Multiple transactions can write to the same extent.
- The head of the rollback segment never moves into a rollback extent currently occupied by the tail.
- Extents in the ring are never skipped over and used out of order as the head tries to advance.
- If the head can't use the next extent, it allocates another extent and inserts it into the ring.

From these principles you can see that transaction time as well as transaction size is important. For instance, a transaction that only modifies one byte but waits a long period of time before ending could cause a rollback segment to grow if the extent it occupied is needed again.

Determining the Proper Rollback Segment Amount and Size

There are two issues that you need to consider when deciding whether your segment is large enough to fit the transaction:

- Make sure that transactions will not cause the head to wrap around too fast and catch the tail. This causes the segment to extend in size, as discussed above.
- If you have long-running queries that access frequently changing data, make sure that the rollback segment doesn't wrap around and prevent the construction of a read-consistent view. (I'll discuss read-consistency issues in more detail in my follow-up 10-Minute Solution, "The ORA-01555 Error: A PL/SQL Developer Solution.")

The size needed for a rollback segment depends directly on the transaction activity of the database. Your primary concern should be the activity during normal processing of the database, not rare or semi-frequent large transactions. Deal with such special cases separately.

The number of rollback segments needed to prevent contention between processes can be determined with the use of the V\$WAITSTAT view. Waits are a definite indication of contention. The following V\$WAITSTAT query displays the number of waits since the instance startup:

```
SELECT Class, Count
FROM V$WAITSTAT
WHERE Class LIKE '%undo%';
```

Any nonzero value in the Count column indicates rollback segment header contention.

The ORA-01555 Error

By Boris Milrud

To find out the size and number of rollback segments needed to handle normal processing on the database, you need to do some testing. A good test is to start with small rollback segments and allow your application to force them to extend. The maximum size that any rollback segment reaches during the test is the size you want to use when configuring segment. If you see any contention, adjust the number of segments and rerun the test. Also, if the largest size requires fewer than 10 extents, or more than 30, it is a good idea to lower or raise the extent size, respectively, and rerun the test. Otherwise, space may be getting wasted during the test, which would throw the number off. For large transactions you can create separate rollback segments.

For sizing rollback segment extents, I strongly recommend that you size each extent equally and make the rollback tablespace a multiple of that extent size. The minimum number of extents for an individual segment should be around 20 for best performance.

Rollback segments dynamically allocate space when required and deallocate space when they are no longer needed (if the OPTIMAL parameter is used). The fewer extents that a rollback segment consists of, the larger and the less granular these space allocations and deallocations are. For example, consider a 200MB rollback segment that consists of only two 100MB extents. If this segment were to require additional space, it would allocate another 100MB extent. This immediately increases the size of the rollback segment by 50 percent and potentially acquires more space than is really needed. By contrast, if the rollback segment consisted of 20 10MB extents, any additional space required would be allocated in 10MB pieces. When a rollback segment consists of 20 or more extents, any single change in the number of extents will not move the total size of the rollback segment by more than 5 percent, resulting in a much smoother allocation and deallocation of space.

Given this, increasing the number of extents beyond the suggested 20 will make space allocation and deallocation even smoother. However, in-house testing shows rapidly diminishing returns when increasing the number of extents past 20. In addition, allocating and deallocating extents is not a cost-free operation. The database will experience performance degradation when performing extent operations. The cost for individual extents is minor but a rollback segment, which is constantly allocating and deallocating tiny extents, can cause even a minor cost to add up.

Ensuring Read Consistency

Oracle always enforces statement-level read consistency. It guarantees that the data returned by a single query is consistent with respect to the time when the query began. Therefore, a query never sees the changes to the data made by transactions that commit during the course of execution of the query.

Oracle uniquely identifies any given point in time by a set of numbers called System Change Numbers. Think of the SCN as the state of the database at any one point in time. As a query enters its execution phase, Oracle assigns a current SCN to it. The query can only see the snapshot of the records as they were at the time they were marked by, or assigned to, the SCN. Oracle uses rollback segments to reconstruct the read-consistent snapshot of the data. Whenever a transaction makes any changes, a snapshot of the record before the changes were made is copied to a rollback segment and the data block header is marked appropriately with the address of the rollback segment block where the changes are recorded. The data block also maintains the SCN of the last committed change to the block.

As the data blocks are read on behalf of the query, only blocks with a lower SCN than the query SCN will be read. If a block has uncommitted changes of other transactions, or has already changed data with a more-recent SCN, then the data will be reconstructed using the saved snapshot from the

The ORA-01555 Error

By Boris Milrud

rollback segments. A rollback segment maintains the snapshot of the changed data as long as the transaction is still active (that is, a commit or rollback has not been issued). Once a transaction is committed, the database marks it with the current SCN and the space used by the snapshot becomes available for reuse. Therefore, an ORA-01555 error will occur if the query is looking for a snapshot that's so old that rollback segment information could not be found because of a wraparound or overwrite.

Next I show you two cases that describe how an ORA-01555 error could be triggered, along with the solution a DBA could follow to avoid the error.

Case 1: An active database with an insufficient number of small-sized rollback segments.

If your database has many transactions changing data and committing changes very often, as in an Online Transactions Processing (OLTP) environment, then the chance of reusing the space used by a committed transaction is higher. A long-running query then may not be able to reconstruct the snapshot due to wraparound and overwrite in rollback segments. Larger rollback segments in this case will reduce the chance of reusing the committed transaction slots.

In this case you should consider adding more rollback segments and increasing their size. The rollback segments' size and number depend on the demands of your application and the number of concurrent users.

Also make sure all rollback segments are online except ones that are reserved for large transactions and are intentionally kept offline. The more segments that are online, the more transactions that are spread out and the less often any individual transaction will be overwritten.

In the rollback storage clause, there is a parameter called OPTIMAL that specify the optimal rollback segment in bytes. When it is set, the database will try to keep the segment at the specified size, rounded up to the extent boundary. If additional space is needed beyond the optimal size, the rollback segment will expand beyond optimal size to accommodate the current transaction(s), but will eventually deallocate extents to shrink back to this size. When the OPTIMAL parameter is set too low, it could lead to frequent ORA-01555 errors because old transaction data may now be eliminated in two ways: by being overwritten or by being discarded during the shrinking process.

To provide best performance, set all of your rollback segments to a size where every single transaction always fits. In practical terms, this may well be impossible—for example, if your largest transaction is 500MB and you require 30 rollback segments for concurrency.

Segments should have an optimal size large enough so that 90 percent or better of transactions will fit without having to extend the segment. In addition, the rollback tablespace should be large enough so that when all rollback segments are at the optimal value, there is plenty of space for them to extend when it becomes necessary. For example, if your segments are set with an optimal value of 50MB and you know that there is a particular transaction that runs infrequently, but requires 1GB when it does run, your rollback tablespace must have at least 950MB free (an absolute minimum) when all segments in that tablespace are at optimal size. Because you cannot normally count on either all segments to be at optimal or the big transaction to be the only one using space in the rollback segment, you should have at least 30 to 40 percent more space available than the absolute minimum.

The ORA-01555 Error

By Boris Milrud

For a batch jobs, data uploads, or any other processes that may contain larger than regular activity transactions, consider creating a large rollback segment and assigning a transaction to it, using the following syntax:

```
COMMIT;  
SET TRANSACTION USE ROLLBACK SEGMENT Rb_Large;
```

Case 2: A rollback segment corruption that prevents a consistent read requested by the query.

If a rollback segment is corrupted and cannot be read, then a statement in the code needing to reconstruct a before-image snapshot will result in the ORA-01555 error. In this case I recommend that you drop and recreate the rollback segment while the database is up and running.

You can drop a rollback segment only if it is offline. To determine whether a rollback segment is offline, query the data dictionary view called DBA_ROLLBACK_SEGS. Offline rollback segments have a STATUS value of AVAILABLE, while offline rollback segments have a STATUS value of IN_USE.

Let's say corruption occurred in rollback segment RBS_02. To put this rollback segment offline and then drop it, execute the following two statements:

```
ALTER ROLLBACK SEGMENT RBS_02 OFFLINE;  
DROP ROLLBACK SEGMENT RBS_02;
```

When you create a rollback segment, it is initially offline and you have to bring it online to make it available for transactions by your Oracle instance. The following statements create a rollback segment RBS_02 with default storage values in the RBS_DATA tablespace and bring it online:

```
CREATE ROLLBACK SEGMENT RBS_02 TABLESPACE RBS_DATA;  
ALTER ROLLBACK SEGMENT RBS_02 ONLINE;
```

You must have ALTER ROLLBACK SEGMENT, DROP ROLLBACK SEGMENT, and CREATE ROLLBACK SEGMENT system privileges to do this. Also, you have to include the TABLESPACE clause in the CREATE ROLLBACK SEGMENT command or else Oracle will create the rollback segment RBS_02 in the SYSTEM tablespace, which Oracle doesn't recommended.

These two cases show how an Oracle DBA can approach the ORA-01555 error problem. The third case, where you fetch across commits while your cursor is open, is covered soon in my follow-up 10-Minute Solution, "The ORA-01555 Error: A PL/SQL Developer Solution."